0014112354 & *Drawing available*
WPI Acc no: 2004-296686/200428
XRPX Acc No: N2004-235678

Control operating method for a motor vehicle's functions, using two interlinked control devices to access data from sensors and to service a computer program for controlling the vehicle's operations

Patent Assignee: BOSCH GMBH ROBERT (BOSC); FINK P (FINK-I); HAFNER A (HAFN-I); ILLG B (ILLG-I); KNOEFLER E (KNOE-I); LANG T (LANG-I); LUNT M (LUNT-I)
Inventor: FINK P; HAFNER A; ILLG B; KNOEFLER E; LANG T; LUNT M; ARNO H; BERND I; ECKEHARD K; MARTIN L; PETER F; TOBIAS L

Patent Family (11 patents, 105 & countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	Type
DE 10343057	A1	20040325	DE 10343057	A	20030916	200428	B
WO 2004027530	A1	20040401	WO 2003EP10285	A	20030916	200431	E
AU 2003273887	A1	20040408	AU 2003273887	A	20030916	200462	E
EP 1543389	A1	20050622	EP 2003757852	A	20030916	200541	E
			WO 2003EP10285	A	20030916		
CN 1647003	A	20050727	CN 2003808676	A	20030916	200577	E
JP 2006515084	W	20060518	WO 2003EP10285	A	20030916	200635	E
			JP 2004537098	A	20030916		
US 20060235594	A1	20061019	WO 2003EP10285	A	20030916	200670	E
			US 2006527601	A	20060424		
EP 1543389	B1	20070418	EP 2003757852	A	20030916	200729	E
			WO 2003EP10285	A	20030916		
DE 50307091	G	20070531	DE 50307091	A	20030916	200736	E
			EP 2003757852	A	20030916		
			WO 2003EP10285	A	20030916		
ES 2283814	T3	20071101	EP 2003757852	A	20030916	200774	E
CN 100380258	C	20080409	CN 2003808676	A	20030916	200845	E

Priority Applications (no., kind, date): DE 10243088 A 20020916

Patent Details

Patent Number	Kind	Lan	Pgs	Draw	Filing Notes	
DE 10343057	A1	DE	23	8		
WO 2004027530	A1	DE				
National Designated States,Original	AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DK DM DZ EC EE EG ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NI NO NZ OM PG PH PL PT RO RU SC SD SE SG SK SL SY TJ TM TN TR TT TZ UA UG US UZ VC VN YU ZA ZM ZW					
Regional Designated States,Original	AT BE BG CH CY CZ DE DK EA EE ES FI FR GB GH GM GR HU IE IT KE LS LU MC MW MZ NL OA PT RO SD SE SI SK SL SZ TR TZ UG ZM ZW					
AU 2003273887	A1	EN			Based on OPI patent	WO 2004027530
EP 1543389	A1	DE			PCT Application	WO 2003EP10285
					Based on OPI patent	WO 2004027530
Regional Designated States,Original	AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LT LU LV MC MK NL PT RO SE SI SK TR					
JP 2006515084	W	JA	30		PCT Application	WO 2003EP10285
					Based on OPI patent	WO 2004027530
US 20060235594	A1	EN			PCT Application	WO 2003EP10285
EP 1543389	B1	DE			PCT Application	WO 2003EP10285
					Based on OPI patent	WO 2004027530
Regional Designated States,Original	AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR					
DE 50307091	G	DE			Application	EP 2003757852
					PCT Application	WO 2003EP10285
					Based on OPI patent	EP 1543389
					Based on OPI patent	WO 2004027530
ES 2283814	T3	ES			Application	EP 2003757852
					Based on OPI patent	EP 1543389

Alerting Abstract DE A1

NOVELTY - Data is processed in a time slice (n) in first (8) and second (9) ten-

millisecond (ms) time slots (81) in sequence, at the beginning of which a master (2)-slave (3) procedure (82) is serviced along with a master-slave transmission procedure (83) for data transmission. In the first time slot, the master's temporary memory holds data from sensors and a current 20-ms time slice.

DESCRIPTION - An INDEPENDENT CLAIM is also included for a computer system with two interlinked control devices joined by a data transmission medium for exchanging synchronizing information between the control devices.

USE - For controlling a motor vehicle's operations like torque, engine speed, fuel injection times, electronic steer by wire, electronic brake by wire, etc.

ADVANTAGE - The control devices exchange synchronizing information. Data transmission is executed via a data transmission medium according to a controller area network protocol.

DESCRIPTION OF DRAWINGS - The drawing shows the different paths of signals in a computer system according to the present invention so as to illustrate data exchange between master and slave in the computer system.

n Time slice
2 Master
3 Slave
8 First 10-millisecond time slot
9 Second 10-millisecond time slot
81 Time slots
82 Master-slave procedure
83 Master-slave transmission procedure

Title Terms /Index Terms/Additional Words: CONTROL; OPERATE; METHOD; MOTOR; FUNCTION; TWO; INTERLINKED; DEVICE; ACCESS; DATA; SENSE; SERVICE; COMPUTER; PROGRAM

Class Codes

International Patent Classification					
IPC	**Class Level**	**Scope**	**Position**	**Status**	**Version Date**
F02D-0041/26	A	I		R	20060101
G05B-0019/042	A	I	F	B	20060101
G05B-0019/042	A	I	F		20060101
G05B-0019/042	A	I		R	20060101
G05B-0019/05	A	I	F	B	20060101
G06F-0017/00	A	I	F	B	20060101
F02D-0041/00	C	I		R	20060101
G05B-0019/04	C	I	F	B	20060101
G05B-0019/04	C	I		B	20060101
G05B-0019/04	C	I		R	20060101
G05B-0019/04	C	I			20060101

ECLA: F02D-041/26D, G05B-019/042M
US Classification, Current Main: 701-048000; Secondary: 701-001000
US Classification, Issued: 70148, 7011

BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen:	102 43 088.8
Anmeldetag:	16. September 2002
Anmelder/Inhaber:	ROBERT BOSCH GMBH, 70469 Stuttgart/DE
Bezeichnung:	Verfahren und Vorrichtung zum Betreiben von miteinander verbundenen Steuergeräten
IPC:	G 05 B, G 08 C

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 6. März 2006
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Wallner

A 9161
06/00
EDV-L

R. 304283

16.09.02 Sy

ROBERT BOSCH GMBH, 70442 Stuttgart

Verfahren und Vorrichtung zum Betreiben von miteinander verbundenen Steuergeräten

Einleitung

Bei einem Master Slave System wird der Motor von mehr als einem Steuergerät gesteuert. Die Steuergeräte werden mit Master und Slave bezeichnet. Das Master Steuergerät beinhaltet mehr Funktionalität als das Slave Steuergerät. Die Steuergeräte sind untereinander mit einem Bussystem zur Datenübertragung vernetzt. Die Sensorik ist entweder an alle Steuergeräte angeschlossen oder nur an den Master. Je nach Einspritz- und Luftsystem werden Sensoren wie Raildruck, Luftmasse, Lufttemperatur, usw. separat an jedes Steuergerät angeschlossen.

Stand der Technik

Bislang existiert kein einheitliches Master Slave Konzept für alle Einspritzsysteme. (Common Rail CRS, Unit Injector UIS)
Je nach Einspritzsystem existieren grundlegende Unterschiede.
Bei CRS wird verschiedene Software in Master und Slave eingesetzt, bei UIS werden keine zylinderindividuellen Mengenberechnungen über den gesamten Kennfeldbereich durchgeführt. Die zeitsynchronen Berechnungen laufen weitgehend unabhängig in den Steuergeräten. Die Sensorik wird nicht abgeglichen für die zeitsynchronen Berechnungen. Der Datenabgleich zwischen den Steuergeräten erfolgt auf funktionaler Ebene. Daher ist das CAN Bus Layout stark abhängig von der "Basisfunktionalität".

Aufgabe der Erfindung

Basisgedanke des erfindungsgemäßen Systems war die Entwicklung eines Konzeptes für alle Einspritzsysteme:

- Master und Slave tauschen ihre Sensorwerte vor dem Beginn jeder zeitsynchronen Berechnung über CAN Bus aus. Alle Hardwaresignale sind in jedem Steuergerät vorhanden, die gekoppelten Zeitscheiben in Master und Slave besitzen gleiche Sensorwerte.
- Kopplung der zeitsynchronen Berechnungen in Master und Slave
- Synchronisation der Regler in Master und Slave
- Unabhängigkeit des Konzeptes vom Einspritzsystem, mit Ausnahme von systemabhängigen Erweiterungen (z.B Raildruckerfassung, BIP-Regelung, UIS-MV Ansteuerung)
- Stabiles CAN Layout bei funktionalen Änderungen
- Gleiche Software in Master und Slave. Die "Rollenverteilung" wird anhand eines Codierpins im Kabelbaum vorgenommen.
- Gleiche Hardware für Master und Slave

R 304283

Aufbau und Funktion

Das entwickelte Master Slave System für das erfindungsgemäße System ist ausgelegt für bis zu vier Steuergeräte, vernetzt über CAN Bus.

Das Systembild in Figur 1 zeigt die in einem Mehr-Zylindermotor eingesetzte Sensorik und Aktorik, sie ist identisch mit der bei dem Konzept eingesetzten, vgl. Figur 2.

Im neu entwickelten Konzept sind Master und Slave Steuergerät baugleich, die eingesetzte Software ist identisch. Beide Steuergeräte sind funktional nahezu identisch, siehe Funktionsauflistung im Systembild. Der Datenaustausch beschränkt sich auf Sensorsignalrohwerte, Digitalsignale, Fehlerstati, Reglerwerte und Ansteuerstatus.

Die Sensorik ist größtenteils am Master angeschlossen. Durch das komplett zweiflutige Luft- und Abgassystem des Motors sind die Signale wie Luftmasse, Lufttemeratur, Ladedruck, Abgastemperatur, usw. an Master und Slave getrennt vorhanden. Gleichzeitig werden aber auch diese doppelten Signale untereinander konsistent ausgetauscht, so dass jedes Steuergerät auch Zugriff auf die Eingangsdaten des jeweils anderen Luftstrangs hat[1]. So können synchron Ersatzreaktionen bei Fehlern eingeleitet werden.
Manche Sensoren, z.B. Glühstiftkerzen, werden nur lokal in jedem Steuergerät erfasst und nicht ausgetauscht, da sie für das Gesamtsystem von untergeordneter Bedeutung sind.

Die zeitsynchronen Berechnungen werden im Master gestartet. Dieser erfasst die entsprechenden Sensorwerte und überträgt einen Teil dieser Sensorrohwerte an den Slave, verwendet aber aus Datenkonsistenzgründen die Sensorwerte der vorangegangenen bereits abgeglichenen Zeitscheibe. Wegen der großen Datenmenge können nicht alle Signale auf einmal übertragen werden. Die Übertragung erfolgt Paketweise (immer 5 oder 6 CAN Botschaften) alle 10 ms vom Master zum Slave und umgekehrt. (vgl. Fig. F).
Ist ein Datenpaket komplett an den Slave übertragen worden, so erkennt der eingesetzte Zustandsautomat zur Synchronisierung der Zeitscheiben im Slave dies und startet die gleiche Zeitscheibe wie der Master. Die Information welche Zeitscheibe gerade gerechnet wird ist in jedem CAN Übertragungspaket enthalten. Die empfangenen Daten werden im ShadowBuffer (siehe dazu Abgleich der Sensorik und Synchronisation der Zeitscheiben im MS System) zwischengespeichert, die aktuelle Zeitscheibe rechnet mit den identischen Daten wie der Master, gespeichert im RamMirror. Grund für die Zwischenspeicherung ist die große Datenmenge einer Zeitscheibe. Ist eine Zeitscheibe komplett zu Ende gerechnet, so muss der Datenaustausch für die nächste Zeitscheibe abgeschlossen sein bevor diese gestartet wird. Durch diesen Zeitaufwand für den konsistenten Datenaustausch wird eine Totzeit im

---

[1] Beim Datenaustausch der doppelten Sensorik wurde außerdem ein Mechanismus entwickelt, mit dem ein Steuergerät auf z.B. die Luftmasse zugreift, und automatisch die am Steuergerät angeschlossene Luftmasse erhält. Über den Zugriff auf die "externe_Luftmasse" wird automatisch auf die Luftmasse des anderen Steuergerätes zugegriffen.

R 304283

System eingebaut, die messbar ist. Die Signale sind alle um eine Zeitscheibe verzögert, aber absolut identisch im System.
Um bereits bei Steuergerätehochlauf die Datenkonsistenz zu gewährleisten wurde eine spezielle Strategie entwickelt, die vor dem eigentlichen Start des Rechnerprogramms ausgewählt Daten abgleicht. Erfreulicher Nebeneffekt dabei ist dass ein ganz spezieller Zeitpunkt im Programm des Master und Slave zeitgleich (im 100er Mikrosekundenbereich) abgearbeitet wird. Dies erlaubt eine Justage der Zeitbasen der einzelnen Steuergeräte. Damit können z.B. Glühkerzensteuersignale in Master und Slave präzise gegeneinander versetzt werden, um Spannungseinbrüche der Batteriespannung zu minimieren.

Neben den Sensorikdaten können auch beliebige Daten aus der Anwendersoftware in diesen Signalaustausch als sogenannte UserSignals einbezogen werden. Diese Signale werden jedoch nicht zwischengespeichert, um deren Aktualität nicht unnötig zu verschlechtern. Abhängig von der Wahl des Übertragungszeitpunktes können auch diese Signale identisch in Master- und Slavezeitscheibe verwendet werden.

Ein Verschieben von Sensoren oder Aktoren an das andere Steuergerät lässt sich mit relativ wenig Konfigurationsänderungsaufwand erledigen, da alle Aktorikwerte in beiden Steuergeräten berechnet werden, und die Endstufenfehler gekoppelt sind.

Auf das Sicherheitskonzept des Master Slave Systems soll an diese Stelle nicht weiter eingegangen werden. Es folgt lediglich eine Auflistung einiger Eckpunkte:

- CAN Bus Hardwareüberwachung
- CAN Bus Datenüberwachung auf Empfang und Versenden
- CAN Bus Datenüberwachung durch Checksumme über komplettes Datenpaket
- Überwachung der korrekten Zeitscheibenabfolge in jedem Steuergerät
- Überwachung der korrekten Zeitscheibenabfolge des Slave durch den Master
- Erkennung eines Steuergeräteresets während des Betriebs durch das andere Steuergerät

Beide Steuergeräte arbeiten das gleiche Programm synchron mit einem einstellbaren Zeitversatz ab, und haben Zugriff auf die gleichen Sensorikdaten. Es handelt sich also eigentlich um ein Master-Master Konzept.

Eine genauere Beschreibung der technischen Abläufe kann den Anlagen entnommen werden.

Die Erfindung ist am Erzeugnis gut nachweisbar. Dazu misst man einfach einen Sensorwert, der beispielsweise nur am Master angeschlossen ist. Der Signalverlauf ist in Master und Slave absolut identisch (abgesehen von der Ausgabeverzögerung durch die Messtechnik).

Gründe für die Konzeptentwicklung

CAN-Layout der vorhergehenden Konzepte stark abhängig von der „Basisfunktionalität", da die M/S CAN Kopplung auf Funktionsebene ist
(CAN Layout muss bei Änderungen immer angepasst werden)

Bei UIS Konzept keine zylinderindividuelle Mengenberechnung über den gesamten Kennfeldbereich

Bei CRS Konzept verschiedene SW in Master und Slave
Unterschiedliche Konzepte für CRS und UIS bei der Vorgängerversion des Systems
Konzepte basieren auf den Randbedingungen der Einspritzsysteme
(z.B. Ansteuerbereich bzgl. OT) => Grundsätzliche Unterschiede der Konzepte

Funktionsaufteilung und Momentenstruktur

Momentenstruktur von Master und Slave gekoppelt
- Gefordertes, inneres Moment des Masters wird übertragen (zeitsynchron)
- n-synch. Regler (LIGov, ASDdc) gekoppelt und vom Master gesteuert
- Master und Slave empfangen CAN Botschaften vom Fahrzeug CAN
- nur Master sendet CAN Botschaften auf Fahrzeug CAN
- FBC wie bei Vorgängerversion des Systems (EDC15)

Zeitsynchrone Momentenberechnung im Slave zur Plausibilisierung
- Dadurch sind im Slave alle Eingangsgrößen für ein zweiflutiges Reglerkonzept vorhanden
- Berechnetes zeitsynchrones Moment im Slave wird nicht für die Zumessung verwendet

Kopplung der Sensorik und Aktorik
- Stabiles CAN Layout bei Änderungen der Basisfunktionalität
- Änderungen nur notwendig wenn zusätzliche Sensoren oder Aktoren notwendig sind

Gleiche SW in Master und Slave (SG Codierpin im Kabelbaum)

Konzept ist unabhängig vom Einspritzsystem
- Systemabhängige Erweiterungen ausgenommen (z.B. Raildruck Erfassung, BIP-Regelung, UIS- MV Ansteuerung)

Kopplung der Sensorik und Aktorik vgl. Figur 3

Zeitsynchrone Berechnungen im Master und Slave über CAN gekoppelt

Master und Slave übertragen ihre Sensorwerte vor Beginn jeder zeitsynchronen Berechnung über CAN

Alle Hardwaresignale (Input / Output) sind in beiden SG somit vorhanden

Aufteilung der 20ms Zeitscheibe in zwei 10ms Zeitscheiben
- Gekoppelte zeitsynchrone Berechnungen besitzen gleiche Sensorwerte

Master und Slave steuern ihre Aktoren separat
- Da die Berechnungen parallel durchgeführt werden, müssen nur die Aktorfehler übertragen werden, nicht die Aktorwerte

Zumessung und Drehzahlerfassung vgl. Figur 4

Drehzahlerfassung für Zylinderhalbsegmente (36°KW Segmente bei 10 Zyl.)

Berechnung der n-sync. Regler bei jedem zweiten Drehzahlsegment mit zwei „36°KW-Drehzahlen" (Optimierung Interruptlast)

Master steuert Parametersätze und Zustand des n-sync. Regler LIGov
- Zusätzlich ist der I-Anteil der LIGov von Master und Slave gekoppelt

Master steuert Parametersätze und Zustand des n-sync. Regler ASDdc

Systemspezifische Zumessung somit unabhängig von M/S Konzept

Master steuert das Moment

Diagnosekonzept

Fehlerspeicherung der Sensoren und Aktoren separat in den Steuergeräten SG

Abgleich der Fehlerspeicherung in Master und Slave möglich

Getrennte Reizadresse für Master und Slave (Master 01, Slave 11)

Funktionale Adressierung für KW2000 geplant

Interner Master Slave CAN (Beispielkonfiguration)

CAN Übertragung mit 1MBit/s

3 winkelsynchrone CAN Objekte von Master zum Slave
- LIGov Steuerung und I-Anteil, ASDdc Steuerung, Endstufenfehler

3 winkelsynchrone CAN Objekte von Slave zum Master
- LIGov I-Anteil, FBC-Menge, Abschaltungen, Endstufenfehler

4 zeitsynchrone CAN Objekte von Master zum Slave im 10ms Raster
- HW Sensorsignale, Aktorfehler

4 zeitsynchrone CAN Objekte von Slave zum Master im 10ms Raster
- HW Sensorsignale, Aktorfehler

1 zeitsynchrones CAN Objekt von Master zum Slave im 20ms Raster
- Momentenkopplung und Überwachung

1 zeitsynchrones CAN Objekt von Slave zum Master im 20ms Raster
- Momentenkopplung und Überwachung

Zur Optimierung des CAN-Layout und der –Auslastung ist es ggf. notwendig die Verteilung der Sensoren auf den Steuergeräten anzupassen

Abgleich der Sensorik und Synchronisation der Zeitscheiben im Master Slave System

(vgl. Figur 5)

Die zeitsynchrone CAN Übertragung

In einem Master Slave System sind zwei Steuergeräte über einen CAN Bus miteinander verbunden. Die Sensoren sind zum Teil nur an einem Steuergerät angeschlossen. Die Sensorwerte müssen somit unter Einhaltung der Datenkonsistenz innerhalb der zeitsynchronen Tasks zum anderen Steuergerät übertragen werden.

Aufbau der CAN Botschaften

Das Daten-Multiplex der CAN Botschaften geschieht mittels DataIDs. Die Botschaften zum Abgleich der Sensorik im Master Slave System haben den folgenden Aufbau:

CANID	DatenID	Daten	Daten	Daten	Daten	Daten	Daten	Daten

Botschaftsübertragung

Die zeitsynchrone Botschaftsübertragung erfolgt im 10 ms Raster. Dabei wird immer ein Block an CAN Botschaften zwischen den Steuergeräten ausgetauscht. Das Blockende und somit das Ende der Datenübertragung wir über einen definierten CAN Identifier erkannt. Dieser CAN Identifier wird exklusiv als letzte Botschaft im CAN Übertragungsblock verschickt.

Aufbau der letzten CAN Botschaft eines Übertragungsblockes:

spezielle CAN ID	DataID	CS	TaskCounter	Daten	Daten	Daten	Daten	Daten

DataID: Mutliplexinformation

CS: Checksumme aller Botschaftsbytes eines Übertragungsblockes

TaskCounter: Schedulinginformation/ Zeitscheibenzähler

Beispiel für einen CAN Übertragungsblock

Sendereihenfolge	CAN Id	Botschaftsinhalt	Multiplex zwischen den 10ms Tasks
1	401	Daten	Ja
2	402	Daten	Ja
3	403	Daten	Ja
LETZTE	404	CS, TaskCounter, Daten	Ja

Botschafts- und Signalpufferung

Um einen korrekten Datenaustausch zwischen Master und Slave zu gewährleisten werden drei verschiedene Signal/ Boschafts-Zwischenspeicher gebraucht:

MS-Message Buffer zur Zwischenspeicherung kompletter CAN Botschaften

Shadow Buffer zur Zwischenspeicherung einzelner Signale, bis alle Signale einer 20ms /100ms Zeitscheibe ausgetauscht sind. Der Zugriff auf den Shadow Buffer erfolgt ausschließlich über die Master Slave Treiber im Rahmen des Botschaftsempfangs, Botschaftsversendens und zum Aktualisieren des RAM Mirrors.

RAM Mirror enthält alle Signale, die für die aktuellen Berechnungen gebraucht werden. Signalzugriffe erfolgen auf diesen Puffer anstatt auf die Hardware (Bsp. A/D Wandler)

Zifferntabelle zu Figur 5

Nr.	Aktion
0	Umkopieren der Signalwerte vom Shadow Buffer in den RAM Mirror
1	Hardwarezugriff mit Zwischenspeicherung des Ergebnisses in Shadow Buffer Rückgabe des im RAM Mirror gespeicherten Signalwertes an die Anwendersoftware
3	CAN bus
4	Abfrage des Signalwertes von der Anwendersoftware
5	Dekodierung der empfangenen CAN Botschaften
6	Zusammenbau der CAN Botschaften zum Versandt

Die zeitsynchronen Tasks (vgl. Figur 6)

Die 20ms und 100ms Task werden in beider Steuergeräten in 10ms Teile zerlegt und im 10ms Zeitraster geschedult. Das heißt, nach 20ms sind alle Teile der 20ms Task und nach 100ms alle Teile der 100ms Task ausgeführt, siehe Figur 6. Das Master Steuergerät sendet alle 10ms CAN Botschaften, um die gemessenen Signalwerte des Master Steuergerätes an das Slave Steuergerät zu transportieren. In dieser CAN Übertragung wird auch die Information welche Zeitscheibe im Master aktuell ausgeführt wird in Form eines Zählers (TaskCounter) mitgesendet. Das Slave Steuergerät synchronisiert den Start seiner 10ms Task mittels eines Zustandsautomaten (siehe). Das Slave Steuergerät überträgt den Zähler für die aktuelle 10ms Zeitscheibe an das Master Steuergerät. Auf diese Weise kann der Master die Abfolge der Zeitscheiben im Slave Steuergerät überwachen.

Strategie zum Datenabgleich während der Initialisierung zwischen Master und Slave (vgl. Figur 3-7)

Problemstellung

Während der Initialaisierung werden Analogwerte gemessen. Mit diesen Werten wird ein PT1-Filter initialisiert.
Bei Master Slave ergibt sich das Problem, dass die Komponententreiber (CD) auf im RAM gepufferte Werte (RAM Mirror) zugreifen. Dieser RAM Mirror enthält während der Initialisierung noch keine gültigen Werte. Somit wird das PT1- Filter falsch initialisiert.

Strategie zur Signalerfassung in der Initialisierung

1. Im MS_Initialize_proc wird das Requestbit im Statusregister für alle in DataSets enthaltenen Signale gesetzt. Somit erfolgt auf jeden Fall ein Hardwarezugriff beim ersten Aufruf des CD. Dieser erste gültige Signalwert wird gemäss MS-System im ShadowBuffer gespeichert. Der CD benutzt den nicht korrekten Wert des RAM Mirror.

2. Abgleich der Signale der Initialisierungstask (Task=10): Die Initialisierungstask hat den TaskCounter 10. Sie wird nur bei Steuergeräte Reset ausgeführt. Master und Slave senden die CAN Daten der Initialisierungstask im MSSigini_proc. Dieser Prozess wird am Ende der Initialisierung ausgeführt, um sicher zu stellen, dass alle HW-Zugriffe der Initialisierung ausgeführt wurden, der ShadowBuffer also mit gültigen Werten gefüllt ist. Nach dem Versenden der Daten der Initialisierungtask warten Master und Slave auf den Empfang des Datenpaketes 10. Die maximale Wartezeit kann über ein Label appliziert werden. Die Daten werden sofort nach dem Empfang[1] dekodiert und somit im eigenen ShadowBuffer gespeichert. Jedes Steuergerät hat jetzt alle für die Initialisierung gültigen Signalwerte im Shadowbuffer hinterlegt.
3. Update der Signale der Initialisierungstask: Kopieren der Signalwerte vom ShadowBuffer in den RAM Mirror.
4. Aufruf der init_procs der Signale, die in der Initialisierung ein PT1-Filter initialisieren.
5. DriveMode

Reset des Master im Fahrbetrieb

Führt der Master im Fahrbetrieb einen Reset aus, so durchläuft er die Initialisierung (s.o.) und senden das Datenpaket 10 der Initialisierungtask auf den CAN B. Der Slave empfängt Datenpaket 10. Der zeitsynchrone Schedulprozess MSSched_proc startet keine Task, sondern sendet das Datenpaket 10 auf dem CAN B. Der Master setzt seine Initialisierung nach der Auswertung dieses Datenpakets fort, siehe oben, Punkt 3.

Reset des Slave im Fahrbetrieb

Führt der Slave im Fahrbetrieb einen Reset aus, so durchläuft er die Initialisierung (s.o.) und senden das Datenpaket 10 der Initialisierungtask auf den CAN B. Der Master empfängt Datenpaket 10. Im zeitsynchronen MSCD_Start-Prozess erkennt er die Anforderung des Datenpakets 10 vom Slave und sendet nicht die Daten der momentanen Task, sondern die Daten der Initialisierungstask. Der Slave setzt seine Initialisierung nach der Auswertung dieses Datenpakets fort, siehe oben, Punkt 3.

Konfiguration der Initialisierungstask (TC=10)

Voraussetzung: HWE.2.1 oder höher
Zur Definition der Task 10 muss im ms_conf.h die Task 10 im Enumerator der Tasks angefügt werden. Anschliessend muss in allen Tabellen (Update, Transmit, Receive, Request jeweils für Master und Slaves) eine elfte Zeile angefügt werden. Es können separate DataSets für diese Initialisierungstask definiert werden. Außerdem ist es möglich, bereits definierte DataSets aus anderen Task zu versenden. Es ist zu beachten, dass ein Multiplex der CAN Identifier nicht erlaubt ist. Nur DataSets, die auf verschiedenen CAN Identifiern liegen können miteinander kombiniert werden. Die letzte CAN Botschaft muss den - wie im Fahrbetrieb definierten – LAST CAN ID nutzen (zahlenmäßig höchster CAN Identifier mit definierter Belegung der CS und des TC in der Botschaft)!

Bei Verwendung der Initialisierungstask sind die Kommentare wie folgt zu setzen und zeigen folgenden Ablauf:

---

[1] Dieser Codeteil wird nahezu zeitgleich in Master und Slave durchlaufen. Hier ist eine Synchronisation der Zeitbasen in M und S möglich, Bsp. GSK3.

```
/* Constant containing the number of task control states
                                                    */
/* IF an INIT State is defined in ms_conf.h subtract 1 else do not subtract 1          */
const MS_numTCS_cu8 = MS_MAX_TASKS_E - 1;
/* if NO INIT State is defined in ms_conf.h use the following statement instead        */
/*const MS_numTCS_cu8 = MS_MAX_TASKS_E;*/

/* Constant containing the TaskCounter for dataexchange in initialization */
/* Use this line if a separate HWE Init Task is used */
const uint8 Ms_stInitTask_cu8 = MS_TASKINIT_E;

/* Use the following line instead if NO separate HWE initialization task is used */
/* const uint8 Ms_stInitTask_cu8 = 0 ;
*/
```

Beispiel zum Datenaustausch im Master Slave System

Figur 8 zeigt den Datenaustausch zwischen Master und Slave Steuergerät. Um das Beispiel übersichtlich zu gestalten wurden die folgenden Vereinfachungen gemacht:

Eine CAN Botschaft enthält nur einen Signalvektor (x1,..x5)

Nur eine CAN Botschaft wird in einer 10ms Task gesendet.

Die Inhalte des RAM Mirrors sind nach dem Ausführen des MS Start Prozesses dargestellt

Die Inhalte des Shadow Buffers sind dargestellt nachdem der MS CAN Transmit Prozess ausgeführt wurde.

R304283

Synchronisation der Zeitscheiben (vgl. Fig. 7)

Sache: Für das Mehrzylinderkonzept ist die synchronisation der Zeitscheiben via CAN zwischen Master und Slave Steuergerät gefordert.

vgl. Figur 7

Aufgabe: Nach Initialisierung soll die Synchronität nach. späterstens 10ms hergestellt sein. Der Zeitversatz zwischen Aktivierung der Zeitscheiben in den jeweiligen Steuergeräten soll applizierbar sein. Bei Unterbrechung der CAN Verbindung muß ein Zwangsscheduling im Slave gestartet werden. Bei Wiederaufbau der Verbindung muß die Synchronität binnen 20ms wieder hergestellt sein.

Ergebnisse: Die 20ms und 100ms Task werden in beiden Steuergeräten in 10ms Teile zerlegt und in zehn 10ms Task geschedult. Das heist, nach 20ms sind alle Teile der 20ms Task geschedult und nach 100ms alle Teile der 100ms Task. Das Master Steuergerät sendet alle 10ms CAN-Botschaften um die Messwerte des Master Steuergerätes an das Slave Steuergerät zu transportieren. (siehe dazu auch „Entwicklungsbericht EDC für Mehrzylinder-Motoren (Master/Slave)") Zusätzlich wird hier ein Zähler mitgesendet, der die Identifizierung der im Master Steuergerät gestarteten 10ms Task sicherstellt. Das Slave Steuergerät synchronisiert mittels einer State-maschine (im inneren dieses Dokumentes näher beschrieben) seine 10ms Task sowohl zeitlich als auch mit der richtigen Identifizierung. Somit ist sichergestellt, dass auch die 20ms und die 100ms Task synchronisiert wird.

R304283

Beschreibung der verwendeten Label zu Fig. 7

MSsched_PerN_C
// normal periode
Normale Periodenlänge, hier 10ms. Wenn das System stabil läuft ist dies die gewünschte Periodendauer im Slave Steuergerät.

MSsched_PerL_C
// long periode
Verlängerte Periodendauer. Wenn öfters (siehe debouncecounter) die Botschaften des Masters später als erwartet im Slave Steuergerät empfangen werden, wird einmalig eine verlängerte Periodendauer eingestellt. Die Länge dieser Periodendauer wird mit diesem Label appliziert.

MSsched_PerS_C
// short periode
Verkürzte Periodendauer. Wenn öfters (siehe debouncecounter) die Botschaften des Master Steuergerätes früher als erwartet empfangen werden, wird einmalig eine verkürzte Periodendauer laut diesem Applikationslabel eingestellt.

MSsched_ShiftL_C
// debouncecounter until periode gets long
Entprellzähler für den Einsatz der verlängerten Periode

MSsched_ShiftS_C
// debouncecounter until periode gets short
Entprellzähler für den Einsatz der verkürzten Periode

MSsched_LockN_C
// Locktime at normal periodeduration
Sperrzeit nach normalem Scheduling. Wenn in diesem Zeitfenster eine gültige Botschaft vom Master empfangen wird, wird der Enprellzähler zum Starten einer verkürzten Periode dekrementiert. Andernfalls wird der Zähler zurückgesetzt.

MSsched_LockS_C
// Locktime at short periodeduration
Sperrzeit nach Scheduling infolge einer verkürzten Periode. Wenn in diesem Zeitfenster eine gültige Botschaft vom Master empfangen wird, wird der Enprellzähler zum Starten einer weiteren verkürzten Periode dekrementiert.

MSsched_Wait_C
// waitingtime until forced scheduling

Wenn die CAN Botschaft des Masters nach Ablauf der Periode im Slave länger als diese Zeit ausbleibt, kommt es zu einem Zwangsscheduling im Slave.

MSsched_WaitI_C
    // waitingtime initalisation until forced scheduling
In diesem Label kann eingestellt werden, wie lange nach Initialisierung auf die erste CAN Botschaft vom Master gewartet werden soll, bevor im Slave ein Zwangsscheduling stattfinden soll.

5.2.. Beschreibung der Meßpunkte

Meßpunkte im Slave gültig:

MSsched_stSync - Synchronisationstatus Master-Slave
    MS_NORM   0x01 -period normal length
    MS_SHORT   0x02 -period short
    MS_WAIT   0x04 -wait to CAN-object after periodtimeout
    MS_NORM_F   0x11 -forced schedule
    MS_LONG   0x14 -period long
    MS_INIRCV   0xA0 -Initialisation oder Recovery

MSSched_MS_NewObj_mp - Merker ob ein neues Objekt empfangen wurde
MSSched_MS_LstObj_mp - zu startende Id der nächsten 10ms Task, bei Zwangsscheduling vom Slave selbstständig weitergezählt sonst von der CAN Id übernommen

MSSched_MS_Per_mp - aktueller Periodenzähler
    Im INKA immer nur alle 20ms ausgegeben, daher immer Maximaler Wert.

MSSched_MS_Lock_mp - aktueller Sperrzähler
MSSched_MS_Shift_mp - aktueller Entprellzähler (debouncecounter)
MSSched_MS_Wait_mp - aktuelle Wartezeit bis Zwangsscheduling
MSSched_MS_TaskCtr_mp - letzte empfangene CAN - Id
MSSched_MS_State_mp - aktueller State
MSSched_MS_Err_mp - Fehlerstatus der Statemaschine
    MS_PERCNTN   0x0100 // Periodcounter underflow
    MS_WAITCNTN   0x0200 // Waitcounter underflow
    MS_LOCKCNTN   0x0400 // Lockcounter underflow

MSSched_MS_ctZeit_mp - Tatsächlich aktivierte Zeitscheibe

Meßpunkte im Master gültig:

MSSched_stSchedule_mp - Zähler um die Id der 10ms Blöcke festzulegen
MSSched_MS_ctZeit_mp - Tatsächlich aktivierte Zeitscheibe

Ansprüche

1. Verfahren zum Betreiben von wenigstens zwei miteinander verbundenen Steuergeräten, wobei die Steuergeräte auf Sensorikdaten zugreifen und jeweils wenigstens ein Programm zur Steuerung von Betriebsabläufen, insbesondere bei einem Fahrzeug, abarbeiten und die wenigstens zwei Steuergeräte Synchronisationsinformationen austauschen
dadurch gekennzeichnet, dass
beide Steuergeräte das gleiche Programm synchron mit einem einstellbaren Zeitversatz abarbeiten.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die wenigstens zwei Steuergeräte Zugriff auf die selben Sensorikdaten besitzen.

3. Vorrichtung zum Betreiben von wenigstens zwei miteinander verbundenen Steuergeräten, wobei die Steuergeräte auf Sensorikdaten zugreifen und jeweils wenigstens ein Programm zur Steuerung von Betriebsabläufen, insbesondere bei einem Fahrzeug, abarbeiten und die wenigstens zwei Steuergeräte Synchronisationsinformationen austauschen
dadurch gekennzeichnet, dass
beide Steuergeräte das gleiche Programm synchron mit einem durch Einstellmittel einstellbaren Zeitversatz abarbeiten.

Konzept 1

Soft Ware Funktionen

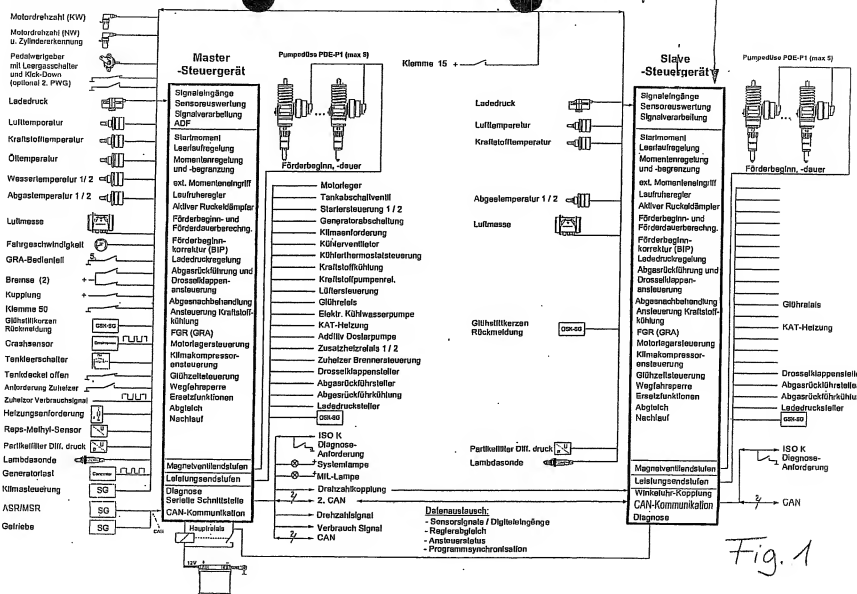


Fig. 1

8/1

Topologie für 10-Zylinder Steuergeräte Master/Slave

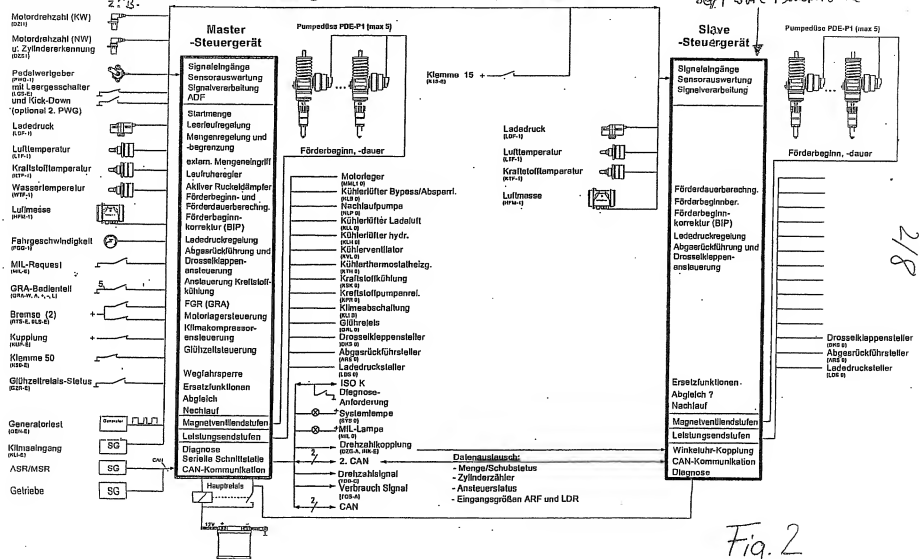


Fig. 2

Darstellung der Zeitsynchronen Übertragung:

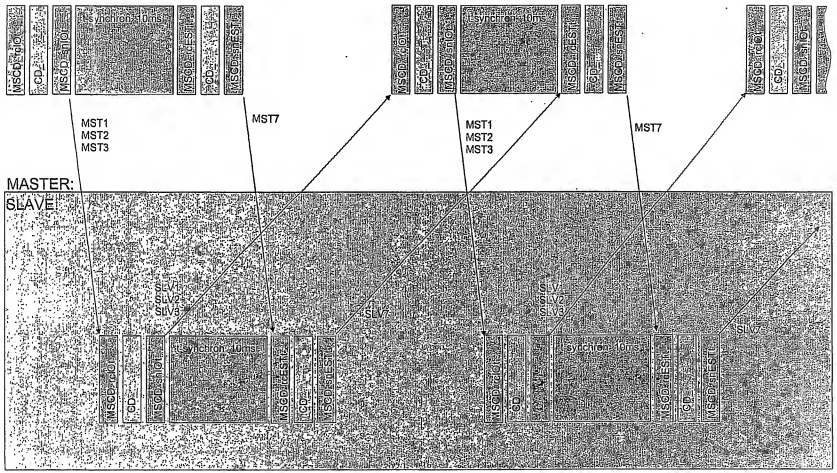
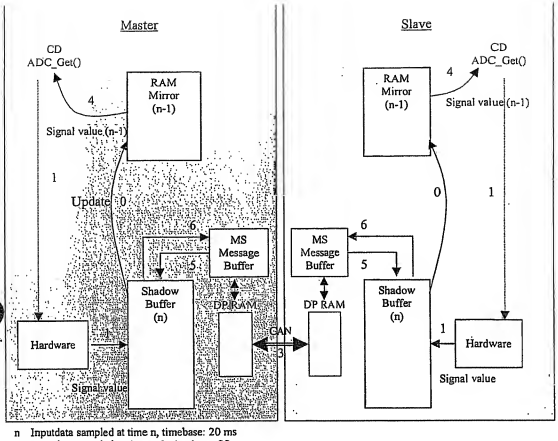


Fig. 3

Darstellung der Winkelsynchronen Übertragung:

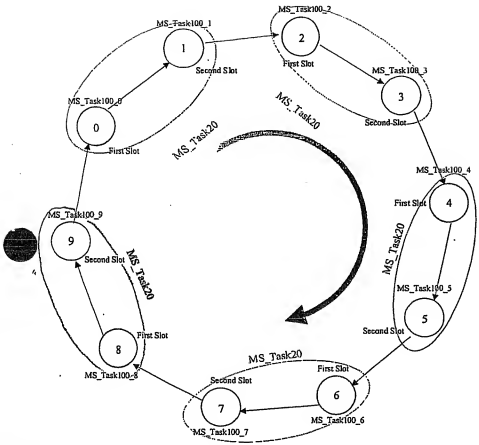
Fig. 4



n Inputdata sampled at time n, timebase: 20 ms
n-1 Inputdata sampled at time n-1, timebase: 20 ms

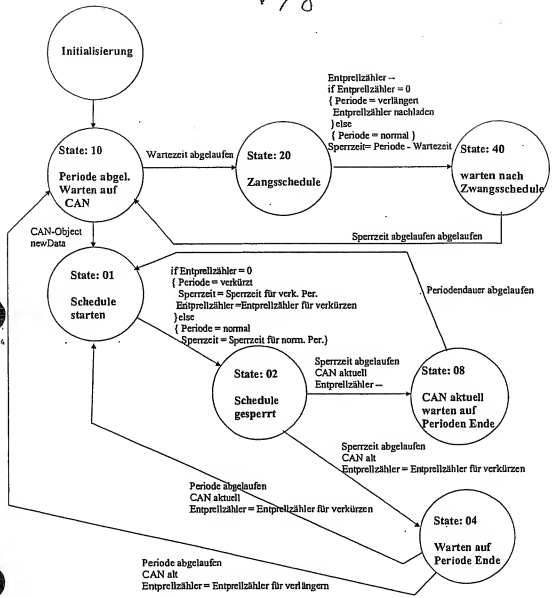
Figur 5 Botschafts- und Signalsspeicherung im Master SlaveSystem

Fig. 5



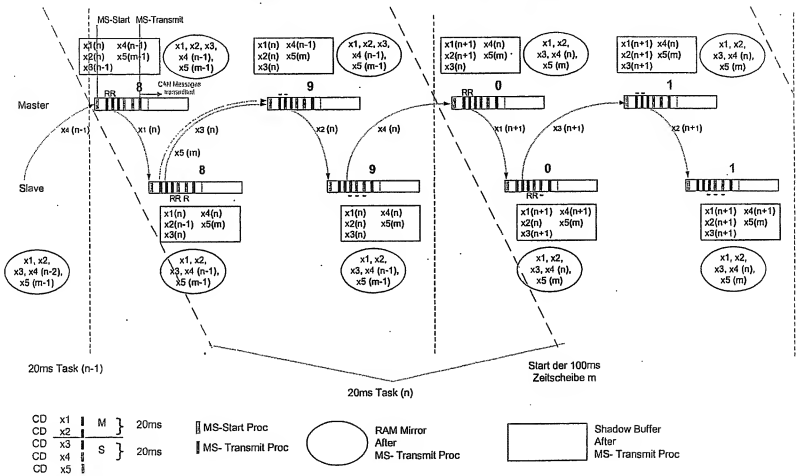
Figur 6 Aufteilung und Abarbeitung der Zeitscheiben

Fig. 6



Figur 7 **Zustandsautomat zur Synchronisierung der Zeitscheiben im Slave**

Fig. 7



Figur 8 Beispiel zum Datenaustausch im Master Slave System

Fig. 8

8/8